WORDSMITHS
Bhavik Shangari - 12240410
Vedant Marodkar - 12240990
Rajeev Goel - 12241460

1 Improving Results with a Lightweight Model Approach on Hivemind

Link

1.1 Introduction

The large parameter sizes and computational costs associated with fine-tuning an 8B parameter model pose a significant challenge, which is done by Mattei et al. A multi-model system called *Hivemind* was developed. This system utilizes multiple LLMs, each fine-tuned for specific subject domains (STEM, Humanities, Social Sciences, Others), and a classifier determines the best model for each input. Despite having such large model the results obtained by a model fine-tuned on an individual domain was achieving the result as same as other model of different domain achieving on same question set.

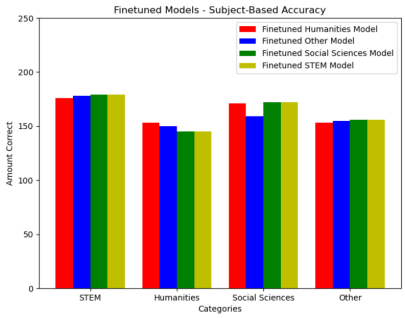


Figure 10: Finetuned Models Subject-Based Accuracy: This figure illustrates how many questions each of the individually finetuned models in our network (one for STEM, one for humanities, one for social sciences, and one for "other") got correct by subject. Note that there were 250 total questions in each category.

The motivation behind our approach is to achieve similar or better results using smaller, more efficient models with fewer parameters, while reducing computational costs. Moreover the model they trained was on only 3000 data points of labeled data. We will be training our model on 99800 points.

1.2 Our Approach

We followed a similar methodology as Hivemind, but instead of fine-tuning four different large models (as in the original study), we fine-tuned a single model on a much larger dataset (99,800 questions without labels). The models we experimented with are:

- Full fine-tuned meta-llama/Llama-3.2-1B-Instruct
- Parameter-efficient version of Llama-3.2-1B-Instruct
- Full fine-tuned meta-llama/Llama-3.2-3B-Instruct
- Parameter-efficient version of Llama-3.2-3B-Instruct

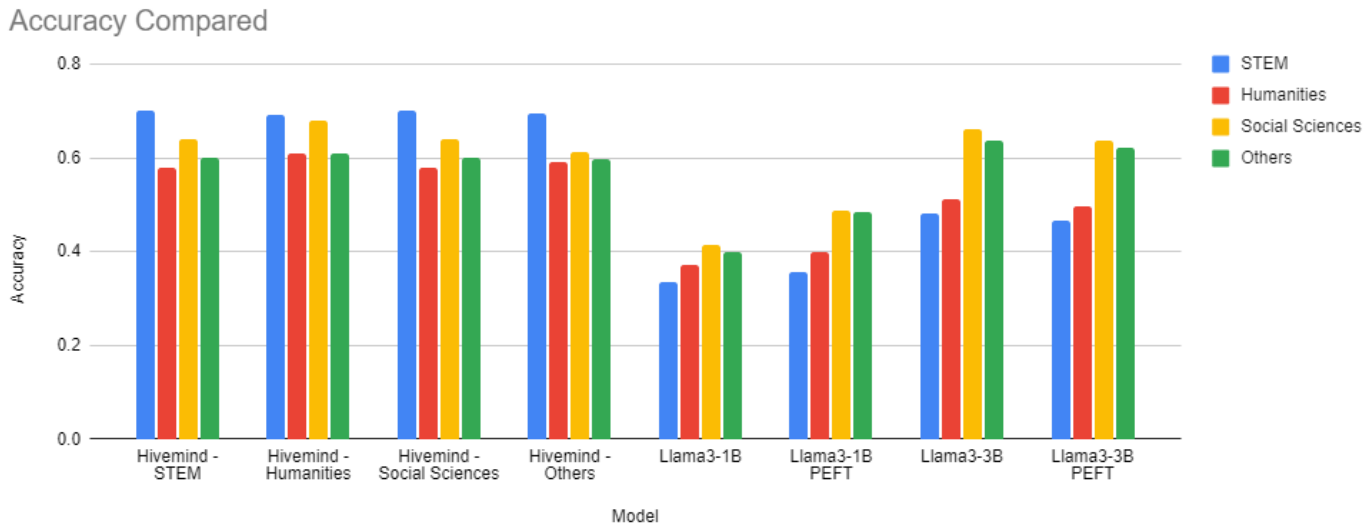
We aimed to improve domain-specific accuracy while reducing computational costs compared to the heavy models in the Hivemind architecture.

1.3 Results

The following table summarizes the results obtained by Mattei et al. and our results for each model across four domains: STEM, Humanities, Social Sciences, and Others.

Model	STEM	Humanities	Social Sciences	Others
Hivemind - STEM	0.7	0.58	0.64	0.6
Hivemind - Humanities	0.692	0.608	0.68	0.608
Hivemind - Social Sciences	0.7	0.58	0.64	0.6
Hivemind - Others	0.696	0.592	0.612	0.596
Llama3-1B	0.3347	0.3719	0.4150	0.3985
Llama3-1B PEFT	0.3575	0.3981	0.4865	0.4843
Llama3-3B	0.480	0.5134	0.662	0.636
Llama3-3B PEFT	0.465	0.496	0.638	0.623

Table 1: Accuracy Results for Different Models and Domains



1.4 Analysis

- The parameter-efficient fine-tuned models performed better for the smaller Llama-1B model, showing improved efficiency for reduced model sizes.
- Larger models such as the Llama-3B exhibited better overall performance, especially in domains such as Social Sciences, with parameter-efficient models performing less effectively at higher model sizes.
- In comparison to the Hivemind approach, our models showed similar performance, with the lighter models performing almost on par with their more resource-intensive counterparts, in domains of Social Sciences and Others.

2 Introducing Dynamic Policy for SMT in Hindi Language

Link

2.1 Introduction

As of now there is no State-of-the-art model that is capable of performing Simultaneous Machine Translation over Indic Languages. However, there are some approaches like Hidden Markov transformer(HMT), SiLLM, Wait-K Policy that satisfies the above objective over foreign languages.
In this phase of the project, we implemented the Wait-K Policy approach for Hindi to English translation.
Moreover, we attempted to achieve better results by implementing our novel approach of POS-SMT Policy.

Wait-K Policy is a static approach that segments the sentence starting with a fixed K(Hyper-parameter) value, and then increasing the window size by 1. Sample seeding array for (k = 5) = [5,6,7,8,9,...]

Issues with this:

- This is unable to handle the redundancy.

- Unnecessary computation in-case of significantly long sentences.

2.2 Our Approach

We implemented our own way POS-SMT Policy. Instead of static policy/seeding array, we considered seeds as the index values where the POS tag of word are:

1. **Noun** or **Verb**
2. Only **Noun**
3. Only **Verb**

Then we compared the results in context of three scoring metrics.

- BLEU
- SacreBLEU
- BERT SCORE

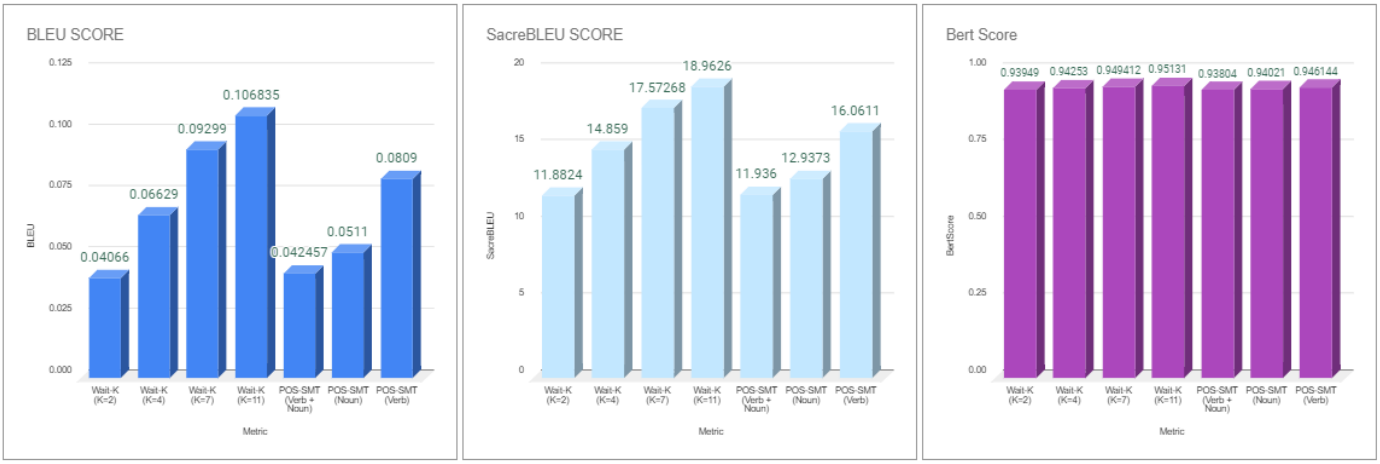
with Wait-k policy implemented with K = 2,4,7 and 11.

2.3 Results

The following table summarizes the results obtained.

Approach	BLEU	SacreBLEU	BertScore
Wait-2	0.04066	11.8824	0.93949
Wait-4	0.06629	14.8590	0.94253
Wait-7	0.09299	17.57268	0.949412
Wait-11	0.106835	18.9626	0.95131
POS-SMT (Verb + Noun)	0.042457	11.9360	0.93804
POS-SMT (Noun)	0.0511	12.9373	0.94021
POS-SMT (Verb)	0.0809	16.0611	0.946144

Table 2: Evaluation metrics (BLEU, SacreBLEU, and BertScore) for different approaches.



2.4 Analysis

- Wait-11 policy turns out to be the best amongst all, although this value of k is practically insignificant for simultaneous machine translation. The optimum value for k as mentioned in paper is between 4 to 6.
- The value of the obtained by **POS-SMT(VERB)** clearly beats the Score values against K=4 and is at par with K=7.
- Hence, this approach of using dynamic seeding is performing better.
- Since, BERT Score of all the models are almost equal, we can say semantically all the models are performing equally.